UNITED STATES PATENT AND TRADEMARK OFFICE

| APPLICATION NO. | FILING DATE | FIRST NAMED INVENTOR | ATTORNEY DOCKET NO. | CONFIRMATION NO. |
|---|---|---|---|---|
| 10/644,399 | 08/19/2003 | Francis X. McKeen | 42P15739 | 7924 |

8791          7590          03/17/2009
BLAKELY SOKOLOFF TAYLOR & ZAFMAN LLP
1279 OAKMEAD PARKWAY
SUNNYVALE, CA 94085-4040

| EXAMINER |
|---|
| GEIB, BENJAMIN P |

| ART UNIT | PAPER NUMBER |
|---|---|
| 2181 | |

| MAIL DATE | DELIVERY MODE |
|---|---|
| 03/17/2009 | PAPER |

**Please find below and/or attached an Office communication concerning this application or proceeding.**

The time period for reply, if any, is set in the attached communication.

PTOL-90A (Rev. 04/07)

| | Application No. | Applicant(s) |
|---|---|---|
| | 10/644,399 | MCKEEN, FRANCIS X. |
| **Office Action Summary** | Examiner | Art Unit |
| | BENJAMIN P. GEIB | 2181 |

*-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --*

## Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE <u>3</u> MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

## Status

1)☒ Responsive to communication(s) filed on <u>30 December 2008</u>.

2a)☒ This action is **FINAL**.          2b)☐ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

## Disposition of Claims

4)☒ Claim(s) <u>1,5-7,9 and 11-19</u> is/are pending in the application.

    4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) <u>1,5,7-9 and 11-19</u> is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

## Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on <u>08 March 2007</u> is/are: a)☒ accepted or b)☐ objected to by the Examiner.

    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

## Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

    a)☐ All   b)☐ Some * c)☐ None of:

      1.☐ Certified copies of the priority documents have been received.

      2.☐ Certified copies of the priority documents have been received in Application No. _____.

      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

    * See the attached detailed Office action for a list of the certified copies not received.

**Attachment(s)**

1)☐ Notice of References Cited (PTO-892)

2)☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3)☐ Information Disclosure Statement(s) (PTO/SB/08)
    Paper No(s)/Mail Date _____.

4)☐ Interview Summary (PTO-413)
    Paper No(s)/Mail Date. _____.

5)☐ Notice of Informal Patent Application

6)☐ Other: _____.

## DETAILED ACTION

### *Specification*

1.      Applicant, via amendment, has overcome the objections to the specification regarding claims 15

and 16 set forth in the previous Office Action.  Consequently, the examiner has withdrawn these

objections.

### *Claim Objections*

2.      Applicant, via amendment, has overcome the objections to claims 9 and 19 set forth in the

previous Office Action.  Consequently, the examiner has withdrawn these objections.

### *Claim Rejections - 35 USC § 101*

3.      Applicant, via amendment, has overcome the 35 USC § 101 rejections to claims 15 and 16 set

forth in the previous Office Action.  Consequently, the examiner has withdrawn these rejections.

### *Claim Rejections - 35 USC § 102*

4.      The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the basis for

the rejections under this section made in this Office action:

> A person shall be entitled to a patent unless –
>
> (e) the invention was described in (1) an application for patent, published under section 122(b), by
> another filed in the United States before the invention by the applicant for patent or (2) a patent
> granted on an application for patent by another filed in the United States before the invention by the
> applicant for patent, except that an international application filed under the treaty defined in section
> 351(a) shall have the effects for purposes of this subsection of an application filed in the United States
> only if the international application designated the United States and was published under Article 21(2)
> of such treaty in the English language.

5.      Claims 1, 5-7, 9, and 11-19 are rejected under 35 U.S.C. 102(e) as being anticipated by Lee et

al., US Patent 6,996,677 (Hereinafter Lee).

6.      Referring to claim 1, Lee has taught a method, comprising:

encountering a function call instruction that calls a called function during program execution *(abstract, column 2, lines 29-50, jump routine)*;

saving a return address in a first stack memory and in a second stack memory at the same time, the return address containing an instruction to be executed after execution of the called function *(abstract, lines 1-12, column 2, lines 29-50, A return address is saved in a first stack memory and a second stack memory upon encountering a jump routine. The first stack memory is element 118 in Figure 2. The second stack memory is shown in Figure 2, elements 110a-110n. 110a-110n is a computer memory consisting of arrays of memory elements stacked one on top of another, which is a stack. See Merriam-webster's online dictionary to support this definition of stack.)*, wherein the second stack memory stores a return address for each function call instruction that calls a called function during program execution *(column 4, lines 6-17, When a subroutine is called during program execution, the return address is stored in element 110a-n.)*;

executing the called function *(abstract, column 2, lines 29-50, A jump to subroutine is executed.)*;

determining if the return address stored in the first stack memory matches the return address stored in the second stack memory to provide protection from a buffer overflow attack *(abstract, column 2, lines 29-50, first comparator and second comparator)*;

executing exception handling code if an exception is generated when the return addresses do not match *(abstract, column 2, lines 29-50, An interrupt signal is generated if the addresses are not the same. Exception handling software is continually executed by the protection co-processor.)*,

wherein the exception handling code determines what value to pass to a program pointer based on the return address retrieved from each of the first and second stack memories *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.)*.

7.      Referring to claim 5, Lee has taught the method of claim 1, as described above, and wherein the exception handling code terminates execution of the program *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, The instruction to move data to the PC register is aborted.)*.

8.      Referring to claim 6, Lee has taught a method, comprising:

          processing instructions within a virtual machine *(abstract, column 2, lines 29-50, column 5, lines 39-41, A software jump/return routine is executed.)*;

          saving a return address in a first stack memory and in a second stack memory at the same time, the return address being an address at which program execution is to resume after execution of a called function *(abstract, column 2, lines 29-50)*;

          comparing the return addresses saved in the first and second stack memories upon execution of the called function *(abstract, column 2, lines 29-50, first comparator)*; and

          exiting the virtual machine if the return addresses do not match to provide protection from a buffer overflow attack *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, The instruction to move data to the PC register is aborted.)* wherein the second stack memory stores a return address for each function call instruction that calls a called function during program execution *(column 4, lines 6-17, When a subroutine is called during program execution, the return address is stored in element 110a-n.)*, and wherein an exception handler determines if the return address from the first stack memory or the return address from the second stack memory is to be used as a value for an instruction pointer *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.)*.

9.      Referring to claim 7, Lee has taught the method of claim 6, as described above, and further comprising passing control to the exception handler *(abstract, column 2, lines 29-50, column 4, line 38-*

*column 5, line 18, column 5, lines 38-41, column 6, lines 59-column 7, line 40, An interrupt signal is*

*generated if the addresses are not the same to load the PC register with a correct value.).*

10.     Referring to claim 9, Lee has taught a method, comprising:

creating first and second stack memories for a program during execution of the program

*(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, Values for the first and*

*second stacks are created and pushed on the stacks during program execution.)*;

encountering a function call to a called function *(abstract, column 2, lines 29-50, column*

*4, line 38-column 5, line 18, jump to subroutine)*;

storing data for the called function and a return address in the first stack memory

*(abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line*

*38-column 5, line 18, first stack)*;

storing the return address in the second stack memory at the same time as the first stack

memory *(abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26,*

*column 4, line 38-column 5, line 18, second stack)*; and

passing control of the program to an exception handler if the return address stored in the

first stack memory does not match the return address stored in the second stack memory upon

execution of the called function to provide protection from a buffer overflow attack *(abstract,*

*column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line 38-*

*column 5, line 18, When the addresses do not match an exception is generated.)*, wherein the

exception handler determines if the return address from the first stack memory or the return

address from the second stack memory is to be used as a value for an instruction pointer

*(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41,*

*column 6, lines 59-column 7, line 40, When the return address values retrieved from both stacks*

*are equal, then the program counter is updated with the current return address value, otherwise*

*an exception is generated to pass the correct value to the program counter.)*.

11.     Referring to claim 11, Lee has taught a processor, comprising:

memory management logic to allocate first and second memory locations corresponding to first and second stack memories, respectively, when a function call instruction calls to a called function is encountered during program execution *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, Values for the first and second stacks are created and pushed on the stacks during program execution of jump to subroutines.)*;

function call logic to write a return address to a memory location from the first memory locations and to a memory location from the second memory locations at the same time *(abstract, lines 19-22, column 2, lines 29-50, A return address is saved in a first stack and a second stack upon encountering a jump routine.)*, the return address being an address at which program flow is to resume after execution of the called function *(abstract, lines 19-22, column 2, lines 29-50, The return address is loaded into the program counter such that program flow resumes after executing a jump instruction.)*; and

buffer overflow control logic to determine if the return address retrieved from the first memory locations matches the return address retrieved from the second memory locations, upon execution of the called function to provide protection from a buffer overflow attack *(abstract, column 2, lines 29-50, first comparator and second comparator)* wherein the exception handler determines if the return address from the first stack memory, or the return address from the second stack  memory is to be used as a value for an instruction pointer *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return address values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.)*.

12.     Referring to claim 12, Lee has taught the processor of claim 11, as described above, and wherein the function call logic and the buffer overflow control logic comprises microcode stored within the processor *(column 5, lines 39-41)*.

13.     Referring to claim 13, Lee has taught a system, comprising:

a memory *(Figure 3, element 102)*; and

a processor coupled to the memory *(Figure 3, at least elements 101, 140, 142, and 144)*, the processor comprising memory management logic to allocate first and second memory locations corresponding to first and second stack memories, respectively, when a function call instruction that calls a called function is encountered during program execution *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, Values for the first and second stacks are created and pushed on the stacks during program execution.)*;

function call logic to write a return address to a memory location from the first memory locations and to a memory location from the second memory locations at the same time *(abstract, lines 19-22, column 2, lines 29-50, A return address is saved in a first stack and a second stack upon encountering a jump routine.)*, the return address being an address at which program flow is to resume after execution of the called function *(abstract, lines 19-22, column 2, lines 29-50, The return address is loaded into the program counter such that program flow resumes after executing a jump instruction.)*; and

buffer overflow control logic to determine if the return address retrieved from the first memory locations matches the return address retrieved from the second memory locations, upon execution of the called function to provide protection from a buffer overflow attack *(abstract, column 2, lines 29-50, first comparator and second comparator)* wherein the exception handler determines if the return address from the first stack memory, or the return address from the second stack memory is to be used as a value for an instruction pointer *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return address values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.)*.

14.     Referring to claim 14, Lee has taught the system of claim 13, as described above, and wherein the memory management logic, the function call logic, and the buffer overflow control logic comprise microcode stored within the processor *(column 5, lines 39-41)*.

15.    Referring to claim 15, Lee has taught a computer recordable type medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

encountering a function call instruction that calls a called function during program execution *(abstract, column 2, lines 29-50, jump routine)*;

saving a return address in a first stack and in a second stack at the same time *(abstract, lines 19-22, column 2, lines 29-50, A return address is saved in a first stack and a second stack upon encountering a jump routine.)*, the return address containing an instruction to be executed after execution of the called function *(abstract, lines 19-22, column 2, lines 29-50, The return address is loaded into the program counter such that program flow resumes after executing a jump instruction.)*;

executing the called function *(abstract, column 2, lines 29-50, A jump to subroutine is executed.)*; and

determining if the return address stored in the first stack matches the return address stored in the second stack to provide protection from a buffer overflow attack *(abstract, column 2, lines 29-50, first comparator and second comparator)* wherein the exception handler determines if the return address from the first stack, or the return address from the second stack is to be used as a value for an instruction pointer *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.)*.

16.    Referring to claim 16, Lee has taught the computer recordable type medium of claim 15, as described above, and wherein the method further comprises generating an exception if the return addresses do not match *(abstract, column 2, lines 29-50, An interrupt signal is generated if the addresses are not the same.)*.

17.     Referring to claim 17, Lee has taught a computer recordable type medium having stored thereon

a sequence of instructions which when executed by a processor, cause the processor to perform a

method comprising:

processing instructions within a virtual machine *(abstract, column 2, lines 29-50, column*

*5, lines 39-41, A software jump/return routine is executed.)*;

saving a return address in a first stack memory and in a second stack memory at the

same time *(abstract, lines 19-22, column 2, lines 29-50, A return address is saved in a first stack*

*and a second stack upon encountering a jump routine.)*, the return address being an address at

which program execution is to resume after execution of a called function *(abstract, lines 19-22,*

*column 2, lines 29-50, The return address is loaded into the program counter such that program*

*flow resumes after executing a jump instruction.)*;

comparing the return addresses saved in the first and second stack memories upon

execution of the called function *(abstract, column 2, lines 29-50, first comparator and second*

*comparator)*; and

exiting the virtual machine if the return addresses do not match to provide protection from

a buffer overflow attack *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18.*

*column 5, lines 38-41, column 6, lines 59-column 7, line 40, The instruction to move data to the*

*PC register is aborted.)* wherein the exception handler determines if the return address from the

first stack memory, or the return address from the second stack memory is to be used as a value

for an instruction pointer *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18.*

*column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return values retrieved*

*from both stacks are equal, then the program counter is updated with the current return address*

*value, otherwise an exception is generated to pass the correct value to the program counter.)*.

18.     Referring to claim 18, Lee has taught the computer recordable type medium of claim 17, as

described above, and wherein the method further comprises passing control to an exception handler

*(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, column 5, lines 38-41, column 6,*

*lines 59-column 7, line 40, An interrupt signal is generated if the addresses are not the same to load the PC register with a correct value.).*

19.     Referring to claim 19, Lee has taught a computer recordable type medium having stored thereon a sequence of instructions which when executed by a processor, cause the processor to perform a method comprising:

creating first and second stack memories for a program during execution of the program *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18, Values for the first and second stacks are created and pushed on the stacks during program execution.)*;

encountering a function call to a called function *(abstract, column 2, lines 29-50, jump routine)*;

storing data for the called function and a return address in the first stack memory *(abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line 38-column 5, line 18, first stack)*;

storing the return address in the second stack memory at the same time as the first stack memory *(abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line 38-column 5, line 18, second stack)*; and

passing control of the program to an exception handler if the return address stored in the first stack memory does not match the return address stored in the second stack memory upon execution of the called function to provide protection from a buffer overflow attack *(abstract, column 2, line 29-column 3, line 15, column 3, line 54-column 4, lines 26, column 4, line 38-column 5, line 18, When the addresses do not match an exception is generated.)* wherein the exception handler determines if the return address from the first stack memory or the return address from the second stack memory is to be used as a value for an instruction pointer *(abstract, column 2, lines 29-50, column 4, line 38-column 5, line 18. column 5, lines 38-41, column 6, lines 59-column 7, line 40, When the return values retrieved from both stacks are equal, then the program counter is updated with the current return address value, otherwise an exception is generated to pass the correct value to the program counter.).*

### *Response to Arguments*

20.    Applicant's arguments filed 12/30/2008 have been fully considered but they are not persuasive.

21.    The applicant argues the rejection/novelty of the claims in substance that:

A) "Lee does not disclose or suggest storing the return address in a first stack memory and a second stack memory at the same time" (page 8)

B) "Lee does not disclose or suggest executing an exception handling code when the return addresses do not match, wherein the exception handling code determines what value to pass a program pointer based on the return address retrieved from each of the first and second stack memories" (page 9)

22.    These arguments are not found persuasive for the following reasons:

Regarding point A), the examiner notes that Lee explicitly states that "whenever a subroutine is called, the content of program counter (PC) is placed in a stack protection data register (SPS data register) 114a-n as well as being placed in the normal fashion into the external memory stack 118." Column 4, lines 6-10. Therefore, the placing of a return address (i.e. the content of a PC) in a first stack memory (i.e. data registers 114) and second stack memory (i.e. external stack memory 118) is done at same time. That is, the address is placed on the stack memories at the point in time that a subroutine is called.

Further regarding point A), the applicant appears to argue that because the SPS data registers 114 are separate from the external stack memory 118 then the storing of the return addresses cannot possibly occur at the same time (remarks; page 8, last paragraph). This is incorrect. Simply because the address is stored at separate locations does not mean that the storage does not occur at the same time. As described above, Lee explicitly states that the storage of return addresses occurs at the same time.

Further regarding point A), the applicant argues that registers 110—110n do not disclose a stack memory. Applicant relies on descriptions within Lee and applicant's specification to show inconsistency with respect to the examiner's interpretation of the word "stack." However, the

examiner notes that the descriptions of within Lee and applicant's specification are not

inconsistent with the examiner's interpretation. Lee describes that a stack is typically comprised

of SRAM or DRAM external to the processor. This is not inconsistent with having an internal

stack made from hardware registers - it is merely a description of what is typical in processors.

Further, Lee repeatedly refers to the stack 118 as the "external memory stack." If stack 118 was

by definition in external memory, then there would be no reason to qualify it as an "external

memory stack." Thus, Lee contemplates a variety of stacks, including internal ones comprised of

hardware registers. Regarding the applicant's asserted definition of a stack memory within the

specification, the examiner notes that page 1, paragraph [0003], is a description of a stack, but

not a definition.

Regarding point B), Lee has taught that when the addresses from an SPS data register

114 and the one from the external memory stack 118 do not match then an interrupt occurs.

Column 4, line 63 - column 5, line 9. As understood in the art, when an interrupt occurs,

exception handling code is executed. Therefore, Lee has taught executing exception handling

code when the return addresses do not match. The exception handling code will inherently have

load the program counter (i.e. pass a value to a program pointer) in order to execute. Because

the exception handling code is executed based on a comparison of the two return addresses, this

program counter load is also performed based on the return addresses.


### *Conclusion*

23.      **THIS ACTION IS MADE FINAL.** Applicant is reminded of the extension of time policy as set forth

in 37 CFR 1.136(a).

A shortened statutory period for reply to this final action is set to expire THREE MONTHS from

the mailing date of this action. In the event a first reply is filed within TWO MONTHS of the mailing date

of this final action and the advisory action is not mailed until after the end of the THREE-MONTH

shortened statutory period, then the shortened statutory period will expire on the date the advisory action

is mailed, and any extension fee pursuant to 37 CFR 1.136(a) will be calculated from the mailing date of

the advisory action.  In no event, however, will the statutory period for reply expire later than SIX

MONTHS from the mailing date of this final action.

Any inquiry concerning this communication or earlier communications from the examiner should

be directed to BENJAMIN P. GEIB whose telephone number is (571)272-8628.  The examiner can

normally be reached on Mon-Fri 8:30am-5:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor,

Alford Kindred can be reached on (571) 272-4037.  The fax phone number for the organization where this

application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application

Information Retrieval (PAIR) system.  Status information for published applications may be obtained from

either Private PAIR or Public PAIR.  Status information for unpublished applications is available through

Private PAIR only.  For more information about the PAIR system, see http://pair-direct.uspto.gov. Should

you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC)

at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative

or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-

1000.

/Alford W. Kindred/                                              Benjamin P Geib
Supervisory Patent Examiner, Art Unit 2181          Examiner
                                                                       Art Unit 2181

/Benjamin P Geib/
Examiner, Art Unit 2181